RUN-TIME LIBRARY TECHNICAL MANUAL

Version 2.1

April 1999

Created by the Center for Applied Informatics
Research and Operations Indianapolis, Indiana

And Distributed by
Department of Veterans Affairs
V*IST***A** Technical Services

Preface

The Run-Time Library package consists of generic routines useful to the development community. It was created for **V***IST***A** Technical Services by the Center for Applied Informatics Research and Operations (CAIRO), Indianapolis, Indiana.

Table of Contents

Introduction

During the course of software development, it is a natural practice to begin to package generic and often needed code into shareable routines. We are certainly no exception to this phenomenon and have created over time a library of useful routines. This document serves to detail the purpose and proper use of each of these routines. We hope you find it and the routines described herein a useful addition to your suite of development tools.

You will note some entry points as marked for deletion. In cases where the Kernel has an equivalent call, we are phasing out our entry points in favor of theirs. We will continue to support the old entry points for at least a year. In the next release (Version 2.2), we will begin moving all routines under the RGUT* namespace for better consistency. This too will be a gradual transition.

Installation

Use the Kernel Installation and Distribution System (KIDS) utility to install this product. See the Run-Time Library V. 2.1 Installation Guide.

User Manual

Since there are no user options and no user interaction with the Run-Time Library, there is no User Manual.

Security Guide

Since there are no files, options, bulletins, mail groups, or menus contained in the Run-Time Library, there is no security guide.

Routines

Routine List

RGCVTDT	RGCVTUU	RGINIT	RGINIT0	RGINIT16
RGINIT18	RGINIT8	RGMSCALR	RGMSCDAT	RGMSCDIC
RGMSCEDT	RGMSCIMP	RGMSCLK2	RGMSCLKP	RGMSCMTL
RGMSCTSK	RGMSCUSR	RGUT	RGUTALR	RGUTBIG
RGUTDAT	RGUTDIC	RGUTDT	RGUTEDT	RGUTFTP
RGUTIMP	RGUTIN	RGUTIN0	RGUTIN16	RGUTIN18
RGUTIN8	RGUTLK2	RGUTLKP	RGUTMTL	RGUTOS
RGUTOS1	RGUTPRE	RGUTRPC	RGUTRRC	RGUTRRT
RGUTSCH	RGUTSRV	RGUTSTX	RGUTSTX0	RGUTSTX1
RGUTTSK	RGUTUSR	RGUTUU	RGXY	RGZOSF
RGZOSF1				

Callable Routines

RGCVTDT

$$^RGCVTDT(RGDAT,RGFMT)

This extrinsic function takes a single argument that is a date/time in either VA FileMan or $HORLOG format. It returns a formatted date/time value. In compliance with Year 2000 (Y2K) requirements, this function no longer supports two-digit year formats.

<return>	A date string formatted according to selected options.
RGDAT	Date/time in either Fileman (e.g., 2930428.1103) or $H (e.g., 53845,234) format.
RGFMT (optional)	Formatting options. Defaults to 0. You may add formatting options occupying different digit positions. Options are: xxxx0 = dd-mmm-yyyy (e.g., 04-Feb-1993) xxxx1 = mmm dd,yyyy (e.g., Feb 04,1993) xxxx2 = mm/dd/yyyy (e.g., 02/04/1993) xxxx3 = mm-dd-yyyy (e.g., 02-04-1993) xxx0x = hh:mm (e.g., 14:33) xxx1x = hh:mm xx (e.g., 02:33 pm) xx0xx = <date> <time> (e.g., 04-Feb-1993 14:33) xx1xx = <date>@<time> (e.g., 04-Feb-1993@14:33) x0xxx = leading zeros (e.g., 04-Feb-1993 02:33 pm) x1xxx = no leading zeros (e.g., 4-Feb-1993 2:33 pm)

RGCVTUU

$$ENCODE^RGCVTUU(X)

Returns the UU-encoded result of X.

$$DECODE^RGCVTUU(X)

Returns the UU-decoded result of X.

RGMSCDAT

NORMAL^RGMSCDAT

This subroutine prompts for a start and end date, returning them in VA FileMan format in the variables RGDAT1 and RGDAT2, respectively. RGDAT2 is guaranteed to be ≥ to RGDAT1. Each of the inputs are saved in ^DISV and may be retrieved by entering a single space at the date prompt.

INVRSE^RGMSCDAT

This subroutine prompts for a start and end date, returning them in VA FileMan inverse format in the variables RGDAT1 and RGDAT2, respectively. RGDAT1 is guaranteed to be ≥ to RGDAT2. Each of the inputs are saved in ^DISV and may be retrieved by entering a single space at the date prompt.

$$ENTRY^RGMSCDAT(%RGP,%RGOPT,%RGDAT,%RGX,%RGY,%RGTRP,%RGHLP)

Prompts for a date and returns it in VA FileMan format. Both of the entry points described above eventually make a call to this extrinsic function. Parameters provide a great deal of control over the behavior of this call as noted below.

<return>	A normal or inverse VA FileMan date, or trapped input (see description of TRP argument).
%RGP (optional)	Prompt to be displayed. If absent or null, defaults to "Enter date: ".
%RGOPT (optional)	Optional option list. The following are valid values: <digit> Identifies input for DISV save/recall F,P,T,X Passed to %DT routine E Uses the line editor for input I Returns date in inverse format
%RGDAT (optional)	Optional default date value.
%RGX (optional)	Screen X-coordinate where prompt will appear. If absent, defaults to the current value of $X.
%RGY (optional)	Screen Y-coordinate where prompt will appear. If absent, defaults to the current value of $Y.

%RGTRP (optional)	Special inputs to trap. If specified, this must contain a valid global or local variable root of a list of inputs that require special handling when encountered. The terminal subscript in the global or local variable is the text to be trapped, while the data is the value to be returned by the routine when the special input is encountered. For example, consider the following list in an arbitrary local variable XYZ: XYZ(“LAST”) = -2 XYZ(“ ALL”)=-3 The variable reference would be passed as “XYZ”. If the user entered LAST or ALL in response to the date prompt, the routine would return values of -2 or -3, respectively. Any other entry would be processed as a normal date.
%RGHLP (optional)	If specified, is the entry point of supplemental help text to be displayed if the user enters a “?” at the date prompt. The format for this argument is “TAG^ROUTINE”. Code at the specified entry point can then display additional help text that will follow the generic help displayed by the routine itself.

RGMSCDIC

$$^RGMSCDIC(%RGDIC,%RGCMD)

Performs one or more operations on entries within a VA FileMan file. This extrinsic function encapsulates and consolidates the VA FileMan API. Although operations are ultimately performed using calls to the VA FileMan API, RGMSCDIC hides much of the complexity of these calls.

RGMSCDIC is built upon the concept of bookmarks. A bookmark contains all of the navigational information required to access the current level within a file. Because the routine handles the construction and parsing of the bookmark, the developer is not required to know anything about its structure other than the fact that the first piece conveys status information about the result of the last operation performed.

Operations upon bookmarks are accomplished with a simple script language. Each script command consists of a single character command code followed by one or more command arguments separated by semicolons. Multiple commands can be passed in a single call by separating them with vertical bars. Each successive command operates on the bookmark produced by the preceding command.

<return>	A bookmark to the current location within the file following execution of all operations specified in %RGCMD. The first piece of the bookmark is a number which can be one of the following: >0 Internal entry number of current file or subfile position 0 No entry is currently selected -1 Requested entry could not be found -2 Requested entry is locked by another process -3 Unexpected error was trapped
%RGDIC	A bookmark or the number or full name of a VA FileMan file.

%RGCMD	String of one or more commands to perform. Separate arguments with a semicolon. Separate multiple commands with a vertical bar. Valid command codes are: ~ Creates a new entry in the current file or subfile. First argument is options to be passed to FILE^DICN in the DIC(0) variable. Second and subsequent arguments are field references as passed in the DIC("DR") variable. The first field reference must be the .01 field of the current level. < Edits the currently selected entry. Format is identical to above with the exception that the first field is not required to be a .01. > Extracts the specified fields from the current level. Field references must be of the format <variable name>=<field number>. Upon return, the specified variables will be loaded with the values of the corresponding fields. = Looks up the argument at the current file or subfile level. + Moves down to the subfile whose internal number is given by the single argument. Note that the current bookmark must reference a valid entry before this operation can be successfully performed. – Moves up one subfile level. This command takes no arguments. @ Deletes the currently selected entry or, if an argument is specified, the entry with the corresponding internal entry number.

Example: S BM=$$^RGMSCDIC(431,"=XYZ|+431.235|~;.01///NEW ENTRY")

This call passes a **V*IST*A** file number (431) as the initial bookmark and perform three commands. The first performs a lookup for entry "XYZ". The second command then moves down a level to subfile 431.235. Finally, a new entry called "NEW ENTRY" is created within the subfile. The returned bookmark would reflect the position of the newly created entry in the subfile. Note that if any command resulted in an error, no subsequent commands would be executed. Also note that each successive command operates upon the bookmark produced by the immediately preceding command.

RGMSCEDT

$$ENTRY^RGMSCEDT(RGDATA,RGLEN,RGX,RGY,RGVALD,RGOPT,RGDISV,RGTERM,RGABRT,RGRM,RGQUIT)

This subroutine presents a simple line editor that permits entry of new text and modification of existing text. The presentation and behavior is much like an edit box in a GUI environment. The subroutine is implemented as an extrinsic function, returning the edited data in its return value.

The parameters for this subroutine are:

<return>	Edited data.
RGDATA (optional)	Initial value of data to edit.
RGLEN (optional)	Maximum length of data. Cannot exceed the maximum string length defined by the operating system (usually 255 bytes). When the edited text reaches this length, further insertions cause the line to be truncated.
RGX (optional)	Initial display X-coordinate. Defaults to the current screen position.
RGY (optional)	Initial display Y-coordinate. Defaults to the current screen position.
RGVALD (optional)	List of valid character inputs. If specified, only characters contained within this string will be accepted.
RGOPT (optional)	Options which may be one or more of the following: E = Suppress echo H = Enable horizontal scrolling I = Suppress input time-out (ignores DTIME) L = Force upper to lowercase O = Overwrite mode Q = Suppress margin bell (quiet mode) R = Use reverse video T = Auto-terminate when character limit reached U = Force lower to uppercase V = Up/down cursor keys terminate input X = Suppress auto-erase feature
RGDISV (optional)	Arbitrary text string that will be used to index previously entered text as saved in ^DISV.
RGTERM (optional)	String of valid input termination characters. By default, only the carriage return character terminates input.

RGABRT (optional)	String of characters that will cause an abort of the input operation. When encountered, the function cancels all changes made to the input data, returning the original value.
RGRM (optional)	Right margin to which input will be restricted. Defaults to the value of the kernel IOM variable if present, or 80 if not.
RGQUIT	Must be passed by reference. Returns an exit code indicating the condition that terminated the routine. Valid return codes are: 1 = A time-out occurred, an abort character was encountered, or the maximum line length (option T only) was reached. 2 = A valid termination character was typed. 3 = The up-arrow was typed (option V only). 4 = The down-arrow was typed (option V only).

The following key combinations have special significance and are intercepted by the subroutine:

Control-A	Toggles insert/replace mode. In insert mode, typed characters are inserted at the cursor position, shifting remaining characters to the right. In replace mode, typed characters overlay existing characters in the same position.
Control-B	Restores edited text to its initial state on subroutine entry.
Control-D	Moves the cursor left one character position.
Control-E	Moves the cursor to the end of the text.
Control-F	Moves the cursor right one character position.
Control-H	Moves the cursor to the beginning of the text.
Control-I	Inserts a space-padded tab. Tab stops are every 8 character positions.
Control-J	Deletes the word at the cursor position.
Control-K	Redisplays the edited text. Useful if the display becomes corrupted by an external event such as a broadcast message.
Control-L	Truncates the line at the cursor position.

Control-R	Restores text deleted by Control-L or Control-U.
Control-U	Deletes text to left of cursor position.
Arrow Keys	Moves the cursor in the respective direction.

RGMSCIMP

$$^RGMSCIMP(RGIMP,RGTRACE)

Imports data from a specially formatted text (host) file into a VA FileMan file.

RGMSCLKP

$$ENTRY^RGMSCLKP(%RGDIC,%RGOPT,%RGPRMPT,%RGXRFS,%RGDATA,%RGSCN,%RGMUL,%RGX,%RGY,%RGSID,%RGRP,%RGHLP)

This subroutine performs lookups on VA FileMan-compatible files. It supports a large number of lookup options. The parameters are:

<return>	The internal identifier of the selected entry or -1 if user entered ^, -2 if user entered ^^, or 0 if user entered null.
%RGDIC	The file number or name of the target **V*IST*A** file or its global root.
%RGOPT (optional)	Lookup options. May contain one or more of the following option codes: A = Automatic selection of an exact match. If the user enters data that exactly matches one and only one existing entry, that item is selected without further intervention. B = Sound the bell each time the selection prompt is displayed. This is useful during data entry to signal that an exact match was not found and further intervention is required to make a selection. D = The cross-references are in date/time format. This permits proper formatting of both the data supplied by the user and the data displayed in the selection list. Note that this option affects all selected cross-references. Thus, you may not have a combination of date/time and non date/time cross-references in the lookup. E = Use the simple line editor when accepting input from the user. This calls the RGMSCEDT line editor. This allows the user to edit the data passed in %D.

%RGOPT (continued)	F = Forget the entry. In the absence of this option code, the lookup routine saves the selected entry in the ^DISV global, making it available for subsequent recall by entering a single space when prompted for input. If this is not desired, this option suppresses this behavior. M = Multiple term lookup and selection allowed. The user is permitted to enter multiple terms separated by semicolons, or make multiple selections from a selection list (also separated by semicolons). The selections are returned in the Y array (see below). O = Show an entry only once. If a file entry is indexed in more than one of the selected cross-references, this option code prevents it from being displayed more than once in the selection list. P = Allow partial lookup. A partial lookup allows the user to enter fragments of a key value, not just the beginning characters of a key value. For example, the entry "EJ F" would match entries "EJECT FCN", "EJECTION FRACTION", and "EJECT FCTN REST". These lookups are less efficient, but often make locating entries much easier for the user. R = Reverse search. Searches cross-references in reverse collating order. This is useful to display date cross-referenences in reverse chronological order. S = Start selection list at previous selection. This starts a selection list at the previous value selected. The previous value must have been selected without the use of the "F" option code. U = Force upper case translation. All user input is translated to upper case before beginning the lookup. 1 = Automatic selection on single match. If the lookup retrieves only one selection, its value is returned without further intervention.

%RGOPT (continued)	* = Search all specified cross-references. In the absence of this option code, the lookup stops at the first cross-reference that produces at least one match. This option forces all cross-references to be searched. ^ = Allow search abort. If the user enters a "^" during a lookup operation, the lookup aborts and any selections located thus far are displayed.
%RGPRMPT (optional)	This is the prompt to be displayed by the lookup routine. If not specified, the prompt "Enter identifier: " is used.
%RGXRFS (optional)	A "^"-delimited list of cross-reference nodes to be searched. If not specified or null, searches all "B"-type cross-references.

%RGDATA (optional)	Data to lookup. If not specified or null, the user is prompted for the information.
%RGSCN (optional)	Names a local or global array which contains screening criteria. Screening criteria control which file entries may be seen by the user. When not specified, all file entries are available for lookup. Otherwise, the named array should contain a list of sequentially numbered entries whose data are M-language expressions that are evaluated against each matched file entry. If any one expression evaluates to nonzero, the entry is accessible to the user. Otherwise, display of the entry is suppressed and the lookup continues. You may reference the variable %S in your expression to refer to the internal identifier of the entry being examined. For example, passing “XYZ” as the value of the SCRN parameter where the named array contains an entry XYZ(1) = "$P(^ZZ(%S,0),U,2)=DUZ" might be used to include only entries belonging to the current user.
%RGMUL	If the multiple selection option is active, the selected entries are returned in the array named in MUL with the subscript being the internal entry numbers of the selected entries.
%RGX (optional)	The X-screen coordinate at which to display the prompt. If not specified, defaults to the value of $X on entry to the routine.
%RGY (optional)	The Y-screen coordinate at which to display the prompt. If not specified, defaults to the value of $Y on entry to the routine.
%RGSID (optional)	Secondary identifier to display. This may either be the field position in the zero node for the entry that is to be displayed or an expression that evaluates to the data to be displayed. When the lookup routine presents a selection list, it presents the selection number in the first column, the index value in the second column, and the optional secondary identifier in the third column. For example, passing "$P(^DPT(%S,0),U,9)" would display the patient's social security number in the third column. This same result could be obtained by passing a value of 9, since the data is in the zero node. Note the use of the variable %S. This variable will always contain the internal entry number of the selection being displayed.
%RGTRP (optional)	Contains the root of a local or global array that specifies special inputs to trap. By specifying an array of special inputs, the lookup routine can identify special cases that are to be returned to the calling routine without further processing. For example, setting XYZ("ALL")="All data" and passing “XYZ” as the value of the TRP parameter would cause the lookup routine to display “All data” as the selection and return the value “ALL” when the user enters "ALL" at the prompt.
%RGHLP (optional)	Routine entry point to display supplemental help text when the user enters a **?** at the lookup prompt. It should be of the form tag^routine. The invoked code may write help text directly to the screen.

RGMSCTSK

$$QUEUE^RGMSCTSK(ZTRTN,ZTDESC,ZTDTH,ZTSAVE,ZTIO,ZTUCI,ZTPRI)

Encapsulates the TaskMan API and simplifies the setup.

<return>	If >0, is the task identifier returned by TaskMan. Any other value signals an error.
ZTRTN	The entry point of the target routine in the format: tag^routine
ZTDESC	Brief description of the task.
ZTDTH (optional)	Date and time task is to begin. This may be either in $H or VA FileMan date formats. If not specified, defaults to the current date and time.
ZTSAVE (optional)	Local variables to be passed to the executing task. These may be passed in either of two ways. If passed by reference, the format is identical to that of the ZTSAVE array in TaskMan. Otherwise, this argument should be a string of ^-delimited variable names. Optionally each variable name can be followed by an equal sign and a value to be assigned.
ZTIO (optional)	Name of the output device to be used by the task. Note that unlike TaskMan, if not specified this defaults to no output device (instead of the current IO device).
ZTUCI (optional)	User Class Identifier (UCI) and or Central Processing Unit (CPU) on which the task is to run. The format should be either UCI:CPU or UCI,CPU.
ZTPRI (optional)	Priority at which the task is to run.

RGUT

$$ASK^RGUT(RGP,RGD,RGZ)

Prompts the user with the text in RGP. This routine expects a Y, N, or ^ response. It returns 1 if the response was Y, 0 if it was N, null if it was ^, or the default value if nothing was entered. An optional default response can be specified in RGD, which is the value returned if nothing was entered (which defaults to N if not specified). If RGZ is passed by reference, it will contain the actual character typed on return.

$$BASE^RGUT(X,Y,L)

This extrinsic function converts a base 10 number in NUM to base BASE, optionally padding (or truncating) to length LEN.

X	Number to be converted. If negative, the function computes the base complement before performing the conversion.
Y	Base to which NUM is to be converted. Y must be greater than 1 and less than 63.
L (optional)	If specified, the result will be left padded with zeroes to this length if shorter or truncated on the right if longer than L.

Example: W $$BASE^RGUT(25,16),!

Displays the number 19 which is 25 base 16.

Example: W $$BASE^RGUT(26,16,4),!

Displays the number 0019.

$$%DT^RGUT(RGD,RGX)

Encapsulates the call to the %DT utility.

$$%DTC^RGUT(X1,X2)

Encapsulates the call to the C^%DTC utility with the added feature that it will support specifying a time component in X1 and fractional days in X2.

$$%DTD^RGUT(X1,X2)

Encapsulates the call to the D^%DTC utility with the added feature that it supports time specifications in

X1 and X2 and will return fractional days.

$$%DTF^RGUT(X)

Converts time specification in X to fractional days.

$$%DTT^RGUT(X)

Converts fractional day specification in X to time.

$$%DTDW^RGUT(X)

Encapsulates the call to the DW^%DTC utility. This entry point will be phased out in favor of the equivalent Kernel call $$DOW^XLFDT.

$$%DTDOW^RGUT(X)

Encapsulates the call to the DOW^%DTC utility. This entry point will be phased out in favor of the equivalent Kernel call $$DOW^XLFDT.

$$%DTNOW^RGUT

Encapsulates the call to the NOW^%DTC utility. This entry point will be phased out in favor of the equivalent Kernel call $$NOW^XLFDT.

$$%DTH^RGUT(X)

Converts a FileMan date/time to $H format. This entry point will be phased out in favor of the equivalent Kernel call $$FMTH^XLFDT.

$$%DTYX^RGUT(X)

Converts a $H date/time to FileMan format. This entry point will be phased out in favor of the equivalent Kernel call $$HTFM^XLFDT.

$$FMTNUM^RGUT(RGNUM)

Formats the numeric value in RGNUM by inserting commas at intervals of every three digits. For example, $$FMTNUM^RGUT(1234567) would return 1,234,567.

$$GBL^RGUT(RGGBL)

Converts a partial global root specification in RGGBL to its normalized form. A normalized global root is suitable for use in expressions that use the @RGGBL@ indirection syntax. This function will convert global specifications such as those used by FileMan (with trailing commas or parentheses), to this normalized form. This entry point will be phased out in favor of the equivalent Kernel call $$CREF^DILF.

$$LOCASE^RGUT(X)

Converts all upper case characters in X to lower case, returning the result. This call will be phased out in favor of the Kernel call $$LOW^XLFSTR.

$$MSG^RGUT(%RGTXT,%RGDLM)

This extrinsic function processes replaceable parameters in a text message, returning the fully instantiated message. A parameter can be any valid M-language expression. If an error is encountered in the evaluation of an imbedded expression the parameter reference is replaced with null.

%RGTXT	The message text which may contain replaceable parameters enclosed by paired delimiters.
%RGDLM (optional)	The parameter delimiter. If not specified, defaults to "%".

Example:
```
S X=1234.5
W $$MSG^RGUT("VALUE IS %X%.")
```

Displays the text: VALUE IS 1234.5.

Example:
```
W $$MSG^RGUT("DATE IS ~$$^RGCVTDT(2940312.1234)~","~")
```

Displays the text: DATE IS 12-Mar-94 12:34

$$RPT^RGUT(X,Y)

Returns the string in X repeated Y times. This call is being phased out in favor of the equivalent Kernel call $$REPEAT^XLFSTR.

$$SET^RGUT(RGCODE,RGSET)

Returns the external form of RGCODE, based on the set of codes described in RGSET. The format of RGSET is identical to that used by FileMan to define a set of codes.

For example: S X=$$SET^RGUT("N","N:NO;Y:YES")

X would contain the value "NO" on return.

$$SNGPLR^RGUT(RGNUM,RGSNG,RGPLR)

Returns singular or plural form of an expression or word depending on the numeric value passed in RGNUM. RGSNG contains the singular form, RGPLR the plural. The value of RGNUM is prepended to the return value.

For example: F X=0,1,2 W $$SNGPLR^RGUT(X,"ITEM","ITEMS"),!

Would produce the following output:

```
NO ITEMS
1 ITEM
2 ITEMS
```

$$SOUNDEX^RGUT(RGVALUE)

Converts the string in RGVALUE to its Soundex equivalent.

$$SSN^RGUT(X)

Converts a 9-digit social security number to its external hyphenated form.

$$STRICMP^RGUT(X,Y)

A part of a popular C function call, this extrinsic makes a case-insensitive comparison of strings X and Y, returning 0 if they are the same, 1 if X collates after Y, and -1 if X collates before Y. Uses the]] operator in its comparison.

$$SUBST^RGUT(RGSTR,RGOLD,RGNEW)

Replaces every occurrence of RGOLD in RGSTR with RGNEW, returning the result.

TITLE^RGUT(RGTTL,RGVER,RGFN)

This entry point clears the screen and displays the title text in RGTTL at the center of the top line of the screen in underlined form. At the left of the top line, the current date is displayed and at the right, the version number (or other text) specified in RGVER. Optionally, a line of informational text (specified in RGFN) can be displayed immediately below the title line.

$$TRIM^RGUT(X,Y)

Strips from X the leading (if Y is -1), trailing (if Y is 1), or leading and trailing (if Y is 0 or missing) spaces, returning the result.

$$TRUNC^RGUT(X,Y)

If the length of the string in X is $\leq$ Y, returns X. Otherwise, returns X truncated to length Y-3 with an ellipsis appended.

$$UFN^RGUT(Y)

Generates a unique filename (8.3 format). If Y is specified, it is used to form the extension of the filename. Otherwise, the extension is randomly generated.

$$UND^RGUT(X)

Generates a string of hyphens, X bytes long. If X is not specified, defaults to the current value of IOM, or to 80 if IOM is not defined. Useful for underlining text.

$$UPCASE^RGUT(X)

Converts all lower case characters in X to upper case, returning the result. This entry point will be phased out in favor of the equivalent Kernel call $$UP^XLFSTR.

$$WORKING^RGUT(X)

Displays a spinning icon indicating an operation in progress. X must be passed by reference and is used to track state information. The function checks to see if the ^ character has been typed, returning true if it has. This can be used as an indicator to the calling process that the current operation should abort.

RGUTRRT

This series of utilities permits the transfer of M-language routines from a host system to remote systems that are connected via Distributed Data Processing(DDP) or Open M Interconnect (OMI) links. This facilitates synchronization of routine partitions across multiple CPU's. These utilities use the ^RGUTL global which must be read and write accessible by all target CPU's. In addition, the global node ^RGUTL("CPUS") should evaluate to a "^"-delimited list of all CPU's that are to be synchronized (e.g., "CSA^CSB^PSA").
Routines to be transferred are copied to a queue in the ^RGUTL global. The utility does not wait until the remote copy operations have completed, but returns immediately. It does verify that the remote copy server routine (^RGUTSRV) is running on all target CPU's, submitting it when it is not found. The remote copy server takes care of installing the routines to its respective CPU. It must be running on each CPU and enters a hibernation state when the routine queue is empty. The last server to install a given routine takes care of removing it from the queue.

^RGUTRRT

When invoked through the main entry point, the user is presented with a routine select prompt. Choose all routines to be transferred. A null entry signals the end of the routine selection.

DELETE^RGUTRRT

This entry point also presents the user with a routine select prompt. Unlike the preceding one, all selected routines are deleted from the current and all remote CPU's. It should be used with great care.

SAVE^RGUTRRT

This entry point submits all routines preloaded in the ^UTILITY global to the routine queue for remote copy or deletion. The format for the ^UTILITY global is:

^UTILITY($J,<routine name>)

If the value of the node is "DELETE", that routine is deleted. Otherwise, it is copied. The programmer is responsible for setting up the ^UTILITY global before the call, and deleting it afterwards.

RRT^RGUTRRT(RGRTN,RGDEL)

This entry point copies or deletes the routine specified in RGRTN to/from the routine queue.

RGRTN	Contains the routine name.
RGDEL (optional)	If present and nonzero, causes the specified routine to be deleted from the host and all remote CPU's. If absent or 0, the specified routine is copied to all remote CPU's.

SHUTDOWN^RGUTRRT

This entry point forces all routine servers to exit.

RGUTSTX

$$ENTRY^RGUTSTX(RGM,RGO)

Performs a syntax check on an M statement.

<return>	Returns 0 if the statement was successfully parsed. Otherwise, returns an error code consisting of three ^-delimited pieces where the first is the numeric error code, the second is the character position where the error was found, and the third is the text of the error message.
RGM	The M statement to be checked.
RGO (optional)	Parsing options. Valid options are: L = Line label allowed • = Dotted syntax allowed N = Do not init parsing tables D = Do not delete parsing tables Z = Process all Z-extensions as valid

RGXY

$$^RGXY(DX,DY)
$$XY^RGUT(DX,DY)

Position the display cursor to the coordinates given by (DX,DY). The upper left display coordinate is always (0,0). The values of $X and $Y are set to the new coordinates. The return value of the extrinsic function is always null.

The entry point $$^RGXY will be phased out in favor of the entry point $$XY^RGUT.

DX (optional)	The X-coordinate value. If not specified, defaults to 0.
DY (optional)	The Y-coordinate value. If not specified, defaults to 0.

Example: W $$XY^RGUT(10,5),"X"

Positions the display cursor at coordinate (10,5) and displays an "X".

Example: S X=$$XY^RGUT

Positions the display cursor at the home position, coordinate (0,0). X will be null upon completion.

RGZOSF

RGZOSF contains a number of calls to support platform-dependent operations such as sequential file input/output and mathematical calculations. Before using any of these utilities, the RGINIT utility must be run once to create the RGZOSF routine appropriate for your M platform. See the installation section for details on how to do this.

CLOSE^RGZOSF(X)

Closes the input/output stream specified in HFS.

X	The I/O stream to be closed. This is the value returned by the *OPEN^RGZOSF* or *$$OPEN^RGZOSF* calls. On some systems it may be the actual filename; on others it may be an HFS port number. If passed by reference, this argument will contain the original filename upon return.

CLOSEALL^RGZOSF

Closes all files opened with the *OPEN^RGZOSF* call.

$$COS^RGZOSF(X)

This entry point will be phased out in favor of the equivalent Kernel call in XLFMTH.

$$CSC^RGZOSF(X)

This entry point will be phased out in favor of the equivalent Kernel call in XLFMTH.

DELETE^RGZOSF(X)

Deletes the file named in X. X must be a valid filename.

X	The name of the file to be deleted. The format will be system-dependent.

DIR^RGZOSF(MASK,MAX,GBL)

Returns a list in the global specified in X3 of up to X2 files matching the file mask in X1.

X1	The file mask used to screen filenames. The specific format will depend upon the underlying operating system and may contain navigational information (i.e., path or directory and device) and wildcard characters.
X2 (optional)	Specifies the maximum number of entries to be retrieved. If not specified or zero, all matching filenames are retrieved.
X3 (optional)	Specifies the global in which to store the returned list of filenames. If not specified, defaults to ^UTILITY("DIR",$J). ***The specified global node is deleted before building the list.***

In the examples that follow, assume that $J evaluates to 10 and the directory being referenced contains the following files:

example.lis
file.txt
noname.bat
sample.txt
test.lis

Example: D DIR^RGZOSF("C:\FILES*.TXT")

Returns the following:

```
^UTILITY("DIR",10,"file.txt")
^UTILITY("DIR",10,"sample.txt")
```

Example: D DIR^RGZOSF("*.*",4)

Returns the following:

```
^UTILITY("DIR",10,"example.lis")
^UTILITY("DIR",10,"file.txt")
^UTILITY("DIR",10,"noname.bat")
^UTILITY("DIR",10,"sample.txt")
```

Note that because a maximum of four files was specified, the fifth file "test.lis" does not appear even though it matches the file mask. Also, not all operating systems return filenames in collating order, so this same file may not be omitted in all instances.

Example: D DIR^RGZOSF("DSA0:[FILES]*.LIS",0,"^TMP($J)")

Returns the following:

```
^TMP(10,"example.lis")
^TMP(10,"test.lis")
```

EOF^RGZOSF

Generates an error if the current I/O device is at the end of file. In some systems, a trappable end-of-file error is automatically generated whenever a read is attempted when at the end of the file. In others, a status variable is set instead. This subroutine permits a consistent approach to detecting and handling end-of-file errors. It should be called prior to performing a read operation. The error trap may determine if the trapped error is an end-of-file error by referencing the extrinsic function $$ISEOF^RGZOSF which will return true if an end-of-file error caused the exception.

ERR^RGZOSF(ERR,ERL)

Returns the current error code in ERR and the associated line label reference in ERL. These arguments must be passed using dot notation. The actual error codes returned are system-dependent.

$$FREE^RGZOSF(X)

Returns the number of megabytes free on the device specified in X.

X	The name of the device to be examined. The format is machine-dependent.

OPEN^RGZOSF(FILE,MODE)

Open the file specified in X1 using the mode specified in X2.

X1	The name of the file to be opened. The specific format will be platform-dependent. It may contain any necessary navigational information (i.e., device name and path or directory). This argument should always be passed using dot notation. Its value upon return will be unchanged on some systems, or will be an HFS port on others. It may then be subsequently used as an argument to the M USE command. Files opened in this manner should be closed with the *CLOSE^RGZOSF* call.

X2 (optional)	The open mode for the file. May be one of three values: R=read-only, W=write (a new file is created), A=append (written data is appended to the end of the file, or a new file is created if one does not exist). The default value is R.

Example:
```
S FIL="TEST.FIL"
D OPEN^RGZOSF(.FIL,"W")
U FIL
W "THIS IS A TEST",!
D CLOSE^RGZOSF(.FIL)
D OPEN^RGZOSF(.FIL)
U FIL
R X
D CLOSE^RGZOSF(.FIL)
```

A file named "TEST.FIL" is created in the default directory and opened in write mode. A single line is written to the file and it is then closed. The file is then reopened in read-only mode. A single line is read from the file and it is once again closed.

$$OPENX^RGZOSF(X1,X2)

Identical to *OPEN^RGZOSF* except that it is implemented as an extrinsic function whose return value is the USE argument to be used for IO.

$$READ^RGZOSF(X,Y)

Reads data from a previously opened HFS device. End-of-file exceptions are handled internally, eliminating the need for a separate exception handler.

<return>	If nonzero, an end-of-file condition was encountered during the read attempt. The return value should always be checked to determine if the read operation was successful.
X	Must be passed by reference. If the read operation is successful, this parameter will contain the data read upon return.
Y (optional)	The host file server device as it is used as an argument to the USE command. If not specified, the routine assumes the currently active IO device is the HFS device.

Example:
```
N HFS,DATA
S HFS=$$OPENX^RGZOSF("TEST.FIL","R")
```

```
F  Q:$$READ^RGZOSF(.DATA,HFS)  D
.U IO(0)
.W DATA,!
D CLOSE^RGZOSF(HFS)
Q
```

An existing file called TEST.FIL is opened for input. An argumentless FOR loop then performs successive reads from the file, outputting the returned data to the home IO device. The loop exits when the end-of-file. The file is closed upon completion.

RENAME^RGZOSF(X1,X2)

Renames the file specified in X1 to the name specified in X2. The filename formats are machine-dependent.

X1	The name of the file to be renamed. The format is machine-dependent.
X2	The new name for the file. The format is machine-dependent. Many systems will not allow files to be renamed to a different directory or device.

$$TEST^RGZOSF(X)

Tests for the presence of the routine named in X or, if X also specifies a label reference in the form tag^routine, tests for the presence of the label in the named routine. Returns true if the reference is found or false otherwise. The circumflex character is optional unless a label reference is specified.

$$VER^RGZOSF

Returns the current version number of the run-time library

.

Internal Relations

Routine	Invokes:
RGCVTDT	$$ENTRY^RGUTDT
RGCVTUU	$$DECODE^RGUTUU,$$ENCODE^RGUTUU
RGINIT	RGUTIN
RGMSCALR	ALERT^RGUTALR,MAIL^RGUTAL,RGMSCDAT, $$ENTRY^RGUTDAT,D1^RGUTDAT,D2^RGUTDAT, INVRSE^RGUTDAT,NORMAL^RGUTDAT
RGMSCDIC	$$ENTRY^RGUTDIC
RGMSCEDT	$$ENTRY^RGUTEDT
RGMSCIMP	$$ENTRY^RGUTIMP
RGMSCLKP	$$ENTRY^RGUTLKP
RGMSCMTL	$$LKP^RGUTMTL,$$PARSE^RGUTMTL, $$PARSE2^RGUTMTL,XREF^RGUTMTL
RGMSCTSK	$$QUEUE^RGUTTSK
RGMSCUSR	ENTRY^RGUTUSR
RGUT	$$^RGCVTDT,$$TRAP^RGZOSF
RGUTALR	ENTRY^RGUTUSR,$$TRAP^RGZOSF
RGUTBIG	$$RPT^RGUT
RGUTDAT	$$^RGCVTDT,$$MSG^RGUT,$$PAUSE^RGUT, $$XY^RGUT,$$ENTRY^RGUTEDT,$$TRAP^RGZOSF
RGUTDIC	$$ENTRY^RGUTLKP,$$TRAP^RGZOSF
RGUTEDT	$$XY^RGUT,$$TRAP^RGZOSF,RAISE^RGZOSF, RM^RGZOSF
RGUTFTP	$$TRAP^RGUTOSCLOSE^RGUTOS,DELETE^RGUTOS, OPEN^RGUTOS,RAISE^RGUTOS,
RGUTIMP	$$TRIM^RGUT,$$TRUNC^RGUT,$$ENTRY^RGUTDIC $$READ^RGZOSF,$$TRAP^RGZOSF,CLOSE^RGZOSF, OPEN^RGZOSF
RGUTIN	$$ENTRY^RGMSCLKP,TITLE^RGUT
RGUTLK2	$$MSG^RGUT,$$TRUNC^RGUT,$$XY^RGUT, $$ENTRY^RGUTDT,$$LKP^RGUTMTL,$$TRAP^RGUT OS
RGUTLKP	$$TRUNC^RGUT,$$XY^RGUT,$$ENTRY^RGUTEDT, $$FMT^RGUTLK2,$$LKP^RGUTLK2,$$VALD^RGUTL K2 DISV^RGUTLK2,HELP1^RGUTLK2,$$TRAP^RGUTOS
RGUTMTL	$$TRAP^RGZOSF,RAISE^RGZOSF
RGUTOS	$$NEWERR^%ZTER,VMS^RGUTFTP
RGUTOS1	$$SUBST^RGUT,$$TRIM^RGUT,VMS^RGUTFTP
RGUTRPC	$$ENTRY^RGUTDIC,$$ENTRY^RGUTSTX
RGUTRRC	TITLE^RGUT,$$TEST^RGZOSF,$$TRAP^RGZOSF
RGUTRRT	TITLE^RGUT,$$QUEUE^RGUTTSK
RGUTSRV	$$UCI^RGUTRRT,$$TRAP^RGZOSF
RGUTSTX	$$LABEL^RGUTSTX0,$$NAME^RGUTSTX0, $$NEXT^RGUTSTX0,CMD^RGUTSTX0,

	$$TRAP^RGZOSF,$$UP^XLFSTR
RGUTSTX0	
	EXP^RGUTSTX1,GLBL^RGUTSTX1,PLIST^RGUTSTX1,
	QT2^RGUTSTX1
RGUTSTX1	$$INT^RGUTSTX0,$$LABEL^RGUTSTX0,
	$$NAME^RGUTSTX0,$$NEXT^RGUTSTX0
	LBL1^RGUTSTX0,LVAL^RGUTSTX0,$$TRAP^RGZOSF
RGUTUSR	$$TRAP^RGZOSF
RGXY	$$XY^RGUT
RGZOSF	$$CVTFN^RGUTOS,$$DIRDLM^RGUTOS,
	$$EOFERR^RGUTOS,$$ETRAP^RGUTOS,
	$$FREE^RGUTOS,$$OPENX^RGUTOS,$$READ^RGUT
	OS,
	$$SIZE^RGUTOS,$$TEST^RGUTOS,$$TRAP^RGUTOS,
	$$VER^RGUTOS,CLOSE^RGUTOS,
	CLOSEALL^RGUTOS,DELETE^RGUTOS,DIR^RGUTOS,
	EOF^RGUTOS,ERR^RGUTOS,FTP^RGUTOS,
	OPEN^RGUTOSRAISE^RGUTOS,RENAME^RGUTOS,
RM^RGUTOS	

Exported Options

There are no Exported Options with the Run-Time Library.

Archiving and Purging

There are no files Exported with the Run-Time Library. Therefore there is no archiving or purging necessary.

External Relations

Run-Time Library requires Kernel Version 8.0 or higher and VA Fileman Version 21.0 or higher. There are no integration agreements.

Package Wide Variables

There are no package wide variables.

Exported Options
Archiving and Purging
External Relations
Package-Wide Variables

Glossary

CPU – Central Processing Unit

DDP – Distributed Data Processing

Extrinsic Function – are written by programmers or they may be supplied by the M vendor. They are not formal language elements. Extrinsic functions are evaluated as expressions and the results of the function replace the call to the function when the code is executed.

OMI – Open M Interconnect